Implementing Secure Multi-Party Computation

Kinan Dak Albab
Boston University
Software and Application Innovation Lab (SAIL)
It is a team effort

Rawane Issa  Andrei Lapets  Peter Flockhart
Mina Michael  Frederick Jansen  Lucy Qin
Rachel Manzelli  Mike Gajda  Mayank Varia
Ben Getchell  Gati Aher  Francis Zamora
What is Secure Multi-Party Computation (MPC)?
Sample Protocol: Sum
Step 1: Secret Sharing

Shares of "50"

One share from each secret
Step 2: Operations
Step 3: Opening The Output

\[
\text{Sum} = 160
\]
Abstract MPC Program

// sharing stage
-45, 75, 50 = share(80)
-45, 56, -583 = exchange(-45, 75, 50)

// operation stage
-572 = sum(-45, 56, -583)

// output stage
160 = open(-572)

// sharing stage
56, 376, -402 = share(30)
376, 75, 722 = exchange(56, 376, -402)

// operation stage
1173 = sum(376, 75, 722)

// output stage
160 = open(1173)

// sharing stage
-583, 722, -89 = share(50)
50, -402, -89 = exchange(-583, 722, -89)

// operation stage
-441 = sum(50, -402, -89)

// output stage
160 = open(-441)
// sharing stage
s1, s2, s3 = share(input)
d1, d2, d3 = exchange(s1, s2, s3)

// operation stage
resultShare = protocol(s1, s2, s3)

// output stage
result = open(resultShare)
var shares = jiff_instance.share(input);
var sum = shares[1];
for (var i = 2; i <= jiff_instance.party_count; i++) {
    sum = sum.sadd(shares[i]);
}

// Return a promise to the final output(s)
return jiff_instance.open(sum);
JIFF: JavaScript Implementation of Federated Functionalities

- Supports general purpose MPC
- Supports 2 or more semi-honest parties
- Secure against coalitions of up to n-1 parties
- Can be used in the pre-processing model
- Ships with several primitives and extensions by default
- Can be easily specialized with different primitive implementations
Yet Another MPC Framework?

Frameworks

- **ABY** - 2PC with secret sharing and garbled circuits; secure against semi-honest adversaries. | NDSS’15.
- **BatchDualEx** - 2PC with garbled circuits; secure against malicious adversaries. | eprint: 2016/632.
- **EMP-toolkit** - 2PC and MPC with garbled circuits; secure against semi-honest adversaries (emp-sh2pc). There are also ones resistant against malicious parties (emp-[ag2pc|m2pc|agmpc]) | eprint: 2017/189, 2016/762, 2017/030.
- **FRESCO** - MPC supporting TinyTables or SPDZ protocols; secure against semi-honest or malicious adversaries. | Practice’15.
- **JIFF** - JavaScript client and server libraries for building web-based applications that employ general purpose MPC; secure against semi-honest adversaries. | documentation: link.
- **MP-SPDZ** - MPC with garbled circuits or secret sharing; secure against malicious or semi-honest adversaries.
- **MPyC** - BGW honest majority multi-party protocol; secure against semi-honest adversaries. | TPMC’18.
- **Obliv-C** - 2PC with garbled circuits; secure against semi-honest adversaries. | eprint: 2015/1153.
- **SCALE-MAMBA** - General MPC with secret sharing; secure against various adversaries including malicious with a dishonest majority. Software closer to a production system. | documentation: link.
- **Sharemind** - 2PC or 3PC with secret sharing; secure against semi-honest adversaries. | Cyber’13.
- **Tf-encrypted** - 3PC with secret sharing; secure against semi-honest adversaries; focused on TensorFlow-based applications.
Why JIFF?
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

- Easy to integrate into Web and Mobile applications
- Built using modern languages and stack compatible with browser, phones, and servers!
- Support parties with intermittent connections, without public IPs, and with variable computing resources
- Must be asynchronous and dynamic
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

- Easy to integrate into Web and Mobile applications
- JavaScript/Node.js
- Built using modern languages and stack compatible with browser, phones, and servers!
- Support parties with intermittent connections, without public IPs, and with variable computing resources
- Must be asynchronous and dynamic
  - No peer-2-peer communication
  - Ability to recover from message loss/crashes
  - Loosely synchronized
  - Easy setup and logistics management
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

Logistics Server

User1

User2

User3
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

Participate in computation as a party or a facilitator!
Informed By Real-World Deployments

Web and Mobile Stack  |  Protocol Design  |  Software Engineering Maturity

- Support flexible roles with flexible trust assumptions
- Support custom asymmetries in MPC layouts and settings
Outsourced MPC

- Many input providers secret share their data with a few compute parties
- The compute parties perform the expensive MPC protocol
- The output is revealed to the input providers, or designated analysts
- Scale: 640,000 shares

Good for reliability and parties with weak computation resources.
Parallel MPC

- Each compute party has several machines/cores/clusters at its disposal
- Computation is carried out by cliques consisting of a machine from each party
- Secret shares are sharded horizontally between cliques
- Applications: Preference Matching, Matrix Multiplication, Search under MPC.
- Scale: 100,000,000 secure multiplications, and half a billion secure additions
  32 machines per party

Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

Good for large data.
Hierarchical MPC

- Parties form a hierarchical layout
- Neighboring parties perform MPC on their data
- Resulting aggregate is passed (in secret shared form) to parties one level up
- Applications: Private accountability for the US court system
- Scale: 100 compute parties
(Full) Asymmetric MPC

- Parties participate in different (overlapping) parts of the MPC protocol
- Parties have different trust assumptions, computing capabilities, and roles
- Parties may learn different information
- Applications: Route recommendation service.
Informed By Real-World Deployments

Web and Mobile Stack | **Protocol Design** | Software Engineering Maturity

Case Study: Binary Search

1 5 8 14 16 19 13
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

Case Study: Binary Search

1 5 8 14 16 19
Case Study: Binary Search

1 5 8 14 16 19

8 < ? = ?
Informed By Real-World Deployments

Case Study: Binary Search

Web and Mobile Stack | **Protocol Design** | Software Engineering Maturity

1 | 5 | 8 | 14 | 16 | 19

8 < 1 = 1
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

Case Study: Binary Search

1 5 8 14 16 19

16 < 0 = 0
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

Case Study: Binary Search

1 5 8 14 16 19

14 < 0 = 0
function binary_search(array, element) {
  while (array.length > 1) {
    // comparison
    var mid = Math.floor(array.length/2);
    var cmp = element.slt(array[mid]);

    // slice
    var nArray = [];
    for (var i = 0; i < mid; i++) {
      var c1 = array[i];
      var c2 = array[mid+i];
      nArray[i] = cmp.if_else(c1, c2);
    }

    // watch out for off by 1 errors if length is odd.
    if (2*mid < array.length) {
      nArray[mid] = array[2*mid];
    }

    array = nArray;
  }

  return array[0].seq(element);
}
Can be done in parallel!

Speculative execution!

Batch multiplication by the same secret!

Can be optimized if the output can be revealed immediately
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

- Easy to customize with low-level primitives or extend with high-level functionalities
- Must be decoupled from any particular assumption or protocols
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

- Easy to customize with low-level primitives or extend with high-level functionalities
- Must be decoupled from any particular assumption or protocols

Provide most general black box protocols built-in
Provide efficient customized protocols as external packages
Informed By Real-World Deployments

Web and Mobile Stack | Protocol Design | Software Engineering Maturity

T. Turban, 2014
Thesis, University of Tartu
What can we do with JIFF?
Pacesetters Deployment

- Two compute parties: a server and an analyst
- 18 input providers grouped into two cohorts for longitudinal analysis
- Analyst initiates the computation, and joins at the end to compute the result
- Input providers can re-submit their data to correct errors
- Secret shares are stored in a DB to survive server outages
- Compute averages and deviations as well as usability metrics
- Collective amount of $ spent analyzed exceeds $100 billion
Applications and Demos

- Data structures: sorting, binary search
- Graph algorithms: minimum spanning tree, shortest path
- Statistics: averages, standard deviations, linear regression, principal component analysis
- SQL-like relational workflows: join, filter, group, aggregate
- Computational Geometry, voting, Battleships!
Where next?
Future Work

- Supporting several logistics server and mixed P2P communication
- Static analysis tools for performance prediction and tradeoff visualizations
Vision - MPC for the Masses

- Building and integrating a collection of tools for simplifying designing and deploying MPC applications
Thank you!

babman@bu.edu

https://github.com/multiparty/jiff